Advanced Regression: 5 Machine learning: Neural networks

Garyfallos Konstantinoudis

Epidemiology and Biostatistics, Imperial College London

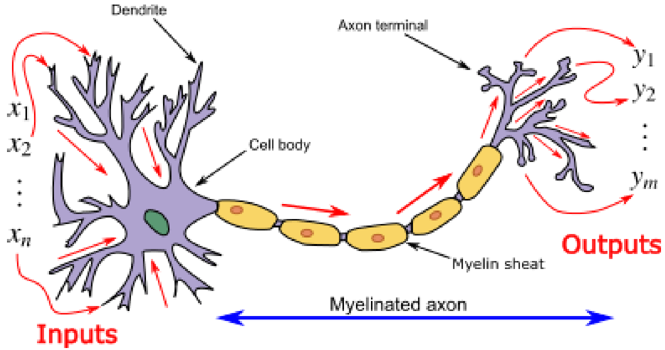
21st March 2023

Week 5 overview

- 10-10:50 Neural Networks: Introduction of basic concepts.
- 11:00-13:00 Tutorial on random forest: Make sure you all do part 1 and discuss it.
- 14:00-15:00 Mock exam & Formative assessment: Go through it together.
- 15:00-15:30 Q&A. Another one 20.04.2023, at 11:00 `https://imperial-ac-uk.zoom.us/j/95015633726?pwd=bDNJTVVoL1Z2ajZzWE0rRGFjYSswZz09`.
- 15:30-16:00 Presentation on lags.
- Course feedback.

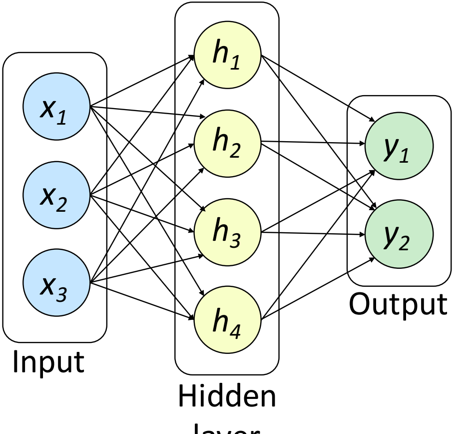
Neural networks

- Neural network are a large class of models and learning methods offering great flexibility.
- Group of units called nodes or neurons that are connected allowing them to transmit signals between themselves.
- Inspired by the brain's interconnected network of neurons.



Neural network's neurons are organised into layers

- **Input**: Predictors or features
- **Hidden layer**: Neurons transmitting signal
- **Output**: Outcome

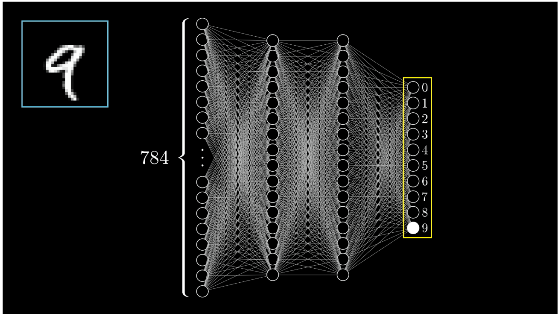


Motivating example: Recognise handwritten digits

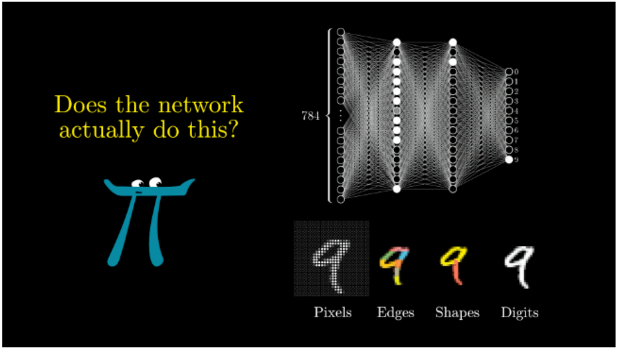
- Online book by Michael Nielsen (`http://neuralnetworksanddeeplearning.com/chap1.html`)

Motivating example: Recognise handwritten digits

- https://www.3blue1brown.com/lessons/neural-networks
- https://www.youtube.com/watch?v=aircAruvnKk



Motivating example: Recognise handwritten digits



Neural networks

K-class classification task:

- y_k, where $k = 1, \ldots, K$ each being coded as a binary dummy variable for the kth class.

- **Input**: p predictors or features $x_1, \ldots, x_p$
- **Hidden layer**: M derived features $h_1, \ldots, h_M$

$$h_m = \sigma(\alpha_{0,m} + \alpha_m^t x), \text{ for } m \in 1, \ldots, M$$

- **Targets**: K predictors

$$t_k = (\beta_{0,k} + \beta_k^t h), \text{ for } k \in 1, \ldots, K$$

- **Output**: K predictions $f_k(x)$

$$f_k(x) = g_k(t), \text{ for } k \in 1, \ldots, K$$

Hidden layer: M derived features $h_1, \ldots, h_M$

$$h_m = \sigma(\alpha_{0,m} + \alpha_m^t x), \text{ for } m \in 1, \ldots, M$$

- $\alpha_{0,m}$ offset or intercept for mth hidden layer
- α_m weights (vector of length p) of the input features
- σ activation functions of $s = \alpha_{0,m} + \alpha_m^t x$:
  - Linear $f(s) = s$
  - Sigmoid

$$f(s) = \frac{1}{1 + \exp(-c \times s)}$$

  - Relu

$$f(s) = \begin{cases} s \text{ if } s \geq 0 \\ 0 \text{ if } s < 0 \end{cases}$$

  - Hyperbolic tangent (Tanh)

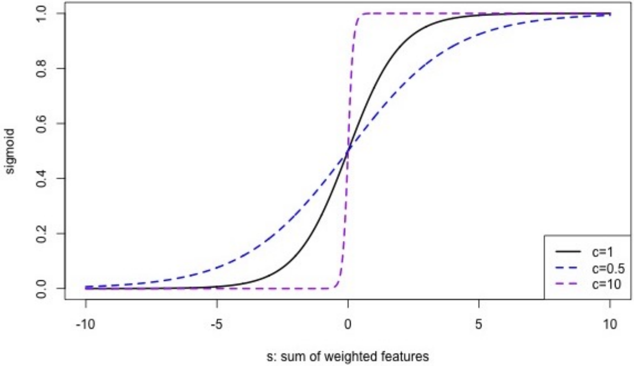
$$f(s) = \frac{\exp(s) - \exp(-s)}{\exp(s) + \exp(-s)}$$

Activation function

Sigmoid function

$$f(s) = \frac{1}{1 + \exp(-c \times s)}$$

- c scale parameter



Prediction function $f_k(x)$

$$f_k(x) = g_k(t),\ k \in 1, ..., K$$

where

- Targets are defined as

$$t_k = (\beta_{0,k} + \beta_k^t h),\ \text{for } k \in 1, ..., K$$

- $\beta_{0,k}$ offset or intercept for kth target
- β_k weights (vector of length m) of the input features
- g_k output function
  - Identity function
  - Softmax

$$g_k(t) = \frac{\exp t_k}{\sum_{k=1}^{K} \exp t_k}$$

Training the neural network

- Feed-forward perceptron which uses only a one hidden layer.
- Parameters to fit:

$$\begin{array}{rcl} \{\alpha_{0,m}, \alpha_m; m \in 1, \ldots M\}; & M(p+1) & \text{feature weights} \\ \{\beta_{0,k}, \beta_k; k \in 1, \ldots K\}; & K(M+1) & \text{prediction weights} \end{array}$$

- Back-propagation algorithm using gradient descent and the chain rule.
- For full details on the algorithm see section 11.4 in Elements of Statistical Learning `https://web.stanford.edu/~hastie/Papers/ESLII.pdf`

Issues in neural networks

- Starting values of the algorithm
- Features need to be scaled prior to the analysis
- Choice of the number of hidden units and layers
  Increasing the amount of hidden layers and units increases the complexity and ability to model more complicated non-linear patterns.
- Overfitting

Regularisation with additional parameters:

- Decay parameter: Penalty for large weights
- Drop out: Random dropping out of nodes during training to prevent overfitting
- Learning rate: Scaling the magnitude of the weight updates

neuralnet function in the neuralnet package

```
neuralnet(f,data=data,hidden,act.fct,linear.output=T)
```

- `f`: formula
- `data`: input data
- `hidden`: specification of the hidden layer structure
- `act.fct`: activation function
- `linear.output=T`: linear activation function

neuralnet function in the neuralnet package

`f`: formula

- `y ~ x1 + x2 + x3`

`hidden`: specification of the hidden layer structure

- Length represents the number of layers and the
- Numbers represent the number of units
- Examples:
  - `hidden = c(5)`: 1 layer with 5 units
  - `hidden = c(5,2)`: 2 layers with 5 and 2 units

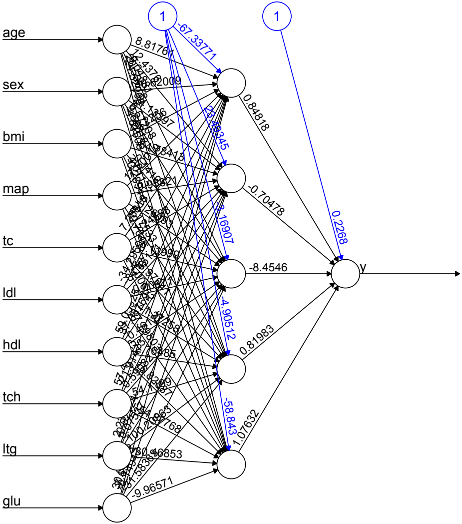
Functionalities:

- `plot.nn`: Plot the neural network.
- `predict.nn`: Prediction of outcome based on new input data.

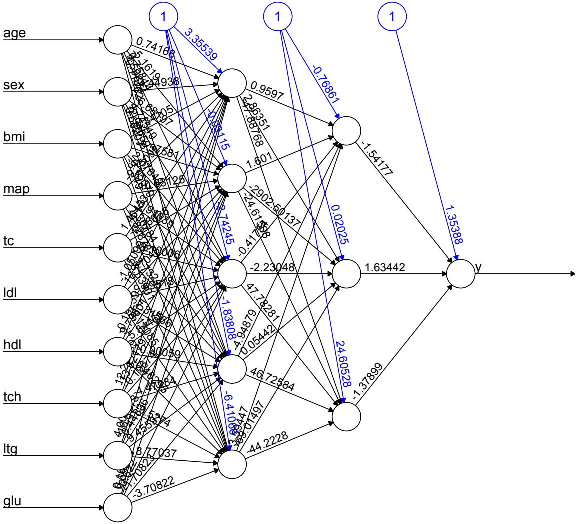
Application example neuralnet

1. Write the formula:
   `f = as.formula(paste("y ~", paste(colnames(x), collapse = "+")))`
2. Fit the neural network with one layer
   `nn1 = neuralnet(f,data=data,hidden=c(5),linear.output=T)`
3. Fit the neural network with two layers
   `nn2 = neuralnet(f,data=data,hidden=c(5,3),linear.output=T)`
4. Plot the neural networks
   - plot(nn1)
   - plot(nn2)

Example neuralnet: One layer with 5 units



Example neuralnet: Two layers with 5 and 3 units



Deep learning in R

Deep learning algorithms

Are neural networks with multiple layers, a deep structure.

- tensorflow package
  Interface to TensorFlow https://www.tensorflow.org, an open source software library for numerical computation using data flow graphs. TensorFlow was developed by the Google Brain Team within Google's Machine Intelligence research organization.
- keras package
  Interface to keras https://keras.io a high-level neural networks application programming interface (API) based on python.

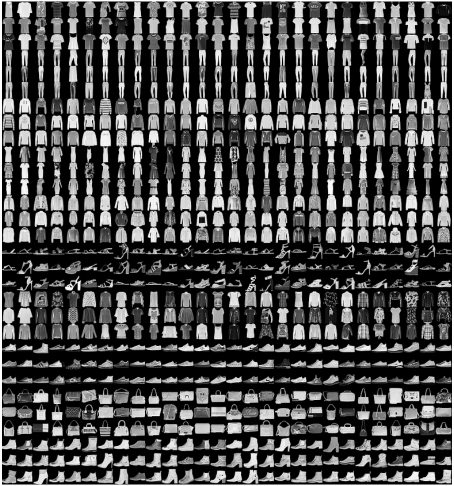
k class classification in keras

- Extensive tutorials online: https://cloud.r-project.org/web/packages/keras/index.html
- Application example from zalando research https://cran.r-project.org/web/packages/keras/vignettes/tutorial_basic_classification.html
- Aim: Predict the product category ($k = 10$ class classification)

0. T-shirt/top
1. Trouser
2. Pullover
3. Dress
4. Coat
5. Sandal
6. Shirt
7. Sneaker
8. Bag
9. Ankle boot

Input

- 70,000 grayscale images at low resolution of 28 by 28 pixels



Data structure

- Training data: 60,000 images

```
> dim(train_images)
[1] 60000    28    28
> table(train_labels)
train_labels
   0    1    2    3    4    5    6    7    8    9
6000 6000 6000 6000 6000 6000 6000 6000 6000 6000
```

- Test data: 10,000 images

```
> dim(test_images)
[1] 10000    28    28
> table(test_labels)
test_labels
   0    1    2    3    4    5    6    7    8    9
1000 1000 1000 1000 1000 1000 1000 1000 1000 1000
```

Building the model

1. **Specify the model**:
```
model = keras_model_sequential()
model % > %
  layer_mflatten(input_shape = c(28, 28)) % > %
  layer_mdense(units = 128, activation = 'relu') % > %
  layer_mdense(units = 10, activation = 'softmax')
```
2. **Compile the model**:
```
model % > % compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)
```
3. **Fit the model**:
```
model % > % fit(train_images, train_labels, epochs =
5)
```

Specify the neural network structure

- `layer_mdense`: Fully connected (dense) hidden layer
- `units`: Number of units
- `activation`: Activation function (exponential, elu, relu, softmax, sigmoid, selu, softplus, tanh and many more)
- Use multiple *layer_mdense* to add more layers

Compile the model

- `optimizer`: Adam, adagrad, adadelta, adamax stochastic gradient descent (SGD) and others
- `loss`: Depending on task (regression: mean_squared_error, mean_absolute_error, huber_loss, kullback_leibler_divergence; classification: categorical_crossentropy, sparse_categorical_crossentropy and many more)
- `metrics`: Depending on task (regression: mean_squared_error, mean_absolute_error; classification: accuracy, binary_accuracy, categorical_accuracy, sparse_categorical_accuracy)

Evaluating training and test error

- **Training error**:

```
score = model %>% evaluate(train_images,
train_labels)
cat('Training loss:', score$loss)
Training loss: 0.2772723
cat('Training accuracy:', score$acc)
Training accuracy: 0.89925
```

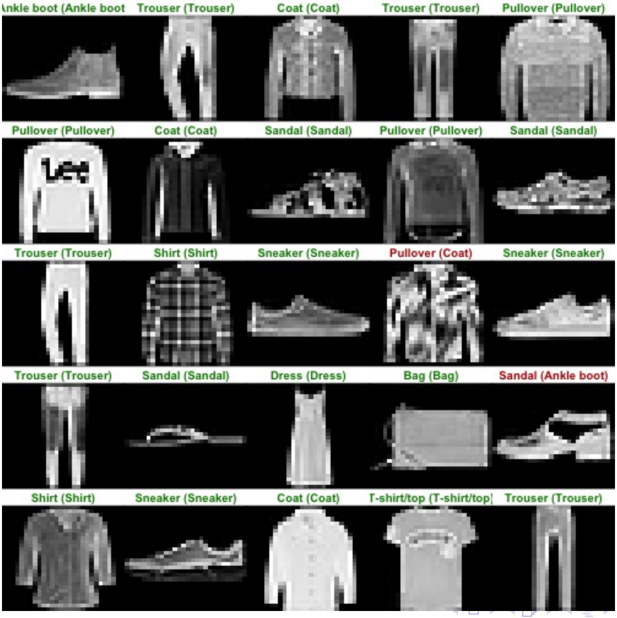
- **Test error**:

```
score = model %>% evaluate(test_images,
test_labels)
cat('Test loss:', score$loss)
Test loss: 0.3530962
cat('Test accuracy:', score$acc)
Test accuracy: 0.8742
```

Making predictions

- Probabilities for class k: `predictions = model %>% predict(test_images)`
- Class: `class_pred = model %>% predict_classes(test_images)`

```
> dim(predictions)
[1] 10000    10
> predictions[1,]
 [1] 2.338658e-06 2.498647e-07 2.077370e-07 1.040700e-09 5.890411e-07
 [6] 9.260748e-03 1.073297e-07 5.394797e-02 2.046366e-05 9.367673e-01
> which.max(predictions[1, ])
[1] 10
> class_pred[1]
[1] 9
> test_labels[1]
[1] 9
```


Ankle boot (Ankle boot
Trouser (Trouser)
Coat (Coat)
Trouser (Trouser)
Pullover (Pullover)
Pullover (Pullover)
Coat (Coat)
Sandal (Sandal)
Pullover (Pullover)
Sandal (Sandal)
Trouser (Trouser)
Shirt (Shirt)
Sneaker (Sneaker)
Pullover (Coat)
Sneaker (Sneaker)
Trouser (Trouser)
Sandal (Sandal)
Dress (Dress)
Bag (Bag)
Sandal (Ankle boot)
Shirt (Shirt)
Sneaker (Sneaker)
Coat (Coat)
T-shirt/top (T-shirt/top)
Trouser (Trouser)

Machine learning resources in R

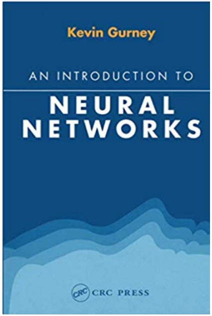
CRAN Task view `https://cran.r-project.org/web/views/MachineLearning.html` presents a collection of established and curated R-packages on

- Neural Networks and Deep Learning
- Recursive Partitioning
- Random Forests
- Regularized and Shrinkage Methods
- Boosting and Gradient Descent
- Support Vector Machines and Kernel Methods
- Bayesian Methods
- Optimization using Genetic Algorithms

Take away: Machine learning: Neural networks and deep learning

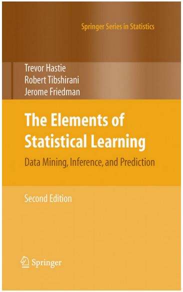
- Neural networks offer a very flexible and powerful framework for prediction.
- Tuning and optimising a neural network involves many parameters (for example number of layers and units, regularisation) and needs to be performed very carefully.
- There are different algorithms for optimisation.
- Neural networks are black boxes which optimise prediction but tell us little about the structure of the data.

Neural network book



- Chapters 1-3
- `https://www.inf.ed.ac.uk/teaching/courses/nlu/assets/reading/Gurney_et_al.pdf`

General background reading
The Elements of Statistical Learning



- Chapter 11
- https://web.stanford.edu/~hastie/ElemStatLearn/